Real-Time Quasi Dense Two-Frames Depth Map for Autonomous Guided Vehicles

André Ducrot, Yann Dumortier, Isabelle Herlin, Vincent Ducrot

Real-Time Quasi Dense Two-Frames Depth Map for Autonomous Guided Vehicles

André DUCROT, Yann DUMORTIER, Isabelle HERLIN, Vincent DUCROT
INRIA centre de Paris Rocquencourt Rocquencourt France

***Abstract*—This paper presents a real-time and dense structure from motion approach, based on an efficient planar parallax motion decomposition, and also proposes several optimizations to improve the optical flow firstly computed. Later, it is estimated using our own GPU implementation of the well-known pyramidal algorithm of Lucas and Kanade. Then, each pair of points previously matched is evaluated according to the spatial continuity constraint provided by the Tensor Voting framework applied in the 4-D joint space of image coordinates and motions. Thus, assuming the ground locally planar, the homography corresponding to its image motion is robustly and quickly estimated using RANSAC on designated well-matched pairwise by the prior Tensor Voting process. Depth map is finally computed from the parallax motion decomposition. The initialization of successive runs is also addressed, providing noticeable enhancement, as well as the hardware integration using the CUDA technology.**

I. Introduction

This contribution addresses the scene perception for intelligent mobile vehicle systems using a monocular video sensor. Perception issues for Autonomous Guided Vehicle consist in estimating the free driving space as well as detecting moving obstacles. Depending on sensors and architecture, such systems can have different price, accuracy and speed characteristics. On one hand, there are active range-finders sensors. Among them, lasers provide high spatial resolutions data at high scanning speeds for a quite expensive price, whereas other cheaper sensors like sonars, are limited to parking applications owing to their range. In addition, the presence of several active sensors in the same area may disrupt measure acquisition. On the other hand, video sensors appear well adapted for automotive applications, due to the rich 2-D information contained in a single image and the 3-D information inferred by two. To date, research has mostly focused on the perception from fix and mobile cameras in static or constrained dynamic scenes [5], [7]. Recovering the 3-D from a monocular sensor is considered as an highly computational task. Most existing approaches do not allow real-time constraints [8] while the others correspond to sparse feature-based methods for Simultaneously Localization And Mapping (SLAM)[9]. In the case of mobile robotic applications, the minimal sufficient information is described by the Extended Minimal Model [10] which includes the road surface estimation and the mobile obstacles tracking. Most feature-based approaches first compute a road displacement model between two frames, and then perform a dense comparison between the second image and the first one, warped using the transformation previously computed to compensate the camera motion. Some important issues inferred from such methods are discussed in section III-A. Recent technological advances, especially regarding the High Performance Computing on common integrated GPU, using CUDA, CTM or OpenCL API, or on dedicated architecture such as FPGA and DSP, allow performing dense motion fields with real-time constraints [3], [11]. Thus Structure From Motion approaches can now be considered to recover the overall depth of an observed scene.

The rest of the document is organized as follows. The section II presents optical flow computation and its real-time integration on GPU, as well as an efficient method to filter the final motion field using the Tensor Voting framework. The section III describes how to identify the ground plane and extract the translational component of the camera's displacement from the image motion. It also details the way to compute a quasi dense depth map from this motion field, in a real-time manner. Then in section IV, we propose two different optimizations to improve the robustness of the depth map estimated in our approach. At last, conclusions and directions for future works are given in section V.

II. Robust Real-Time optical flow

So far, many efforts in computer vision have been focused on retrieving the 3-D location of 2-D image points. Depending of the considered application and its relative constraints, the "inverse projection" of 2-D points in the Euclidean space can be classified in different processes such as:

- Matching sparse points or structures of interest according to some specific descriptors, trackers and spatial constraints.
- Directly estimating the depth map from the epipolar geometry, given the baseline between two or more cameras operating simultaneously, and using multi-views constraints.
- Approximating the image motion field, by using a temporal photometric constraint, that consists in computing the optical flow.

The latter commonly used the brightness constancy constraint, also called optical flow constraint, which assumes that the pixel intensity of an image point does not change with time. It can be written in a discrete manner as follows:

$$I_1(\mathbf{x}_1) = I_2(\mathbf{x}_2) = I_2(\mathbf{x}_1 + \triangle\mathbf{x}_{21}), \quad (1)$$

with $\mathbf{x}_i$ the image coordinates of a point X and $I_j(\mathbf{x}_i)$ the brightness of its pixel at time t_i and t_j, respectively. $\triangle \mathbf{x}_{ji}$ describes the discrete image motion of X between t_i and t_j. In the continuous case, with $dI/dt = 0$, the differential form of (1) is given by:

$$\begin{aligned} \frac{dI}{dt} &= \frac{dI(x(t), y(t), t)}{dt} \\ &= I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t \\ &= \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} dx/dt \\ dy/dt \end{bmatrix} + I_t \\ &= \nabla I^T \boldsymbol{\omega} + I_t \end{aligned} \tag{2}$$

where I_x, I_y and I_t represent the partial spatial and temporal derivatives describing the image gradient ∇I, and $\boldsymbol{\omega}$ the image motion.

A. Real-time Lucas and Kanade tracker

1) Lucas and Kanade: Given any image point x, the Eq. (2) only provides an estimation of its 2-D motion along spatial gradient direction. This issue, well-known as the aperture problem, can partially be solved by integrating the optical flow constraint on the neighborhood Ω_{ROI} of x. In [1] Lucas and Kanade assume the motion uniform over Ω_{ROI} to minimize the following expression:

$$J(\boldsymbol{\omega}) = \sum_{\Omega_{ROI}} (\nabla I \, \boldsymbol{\omega} + I_t)^2 \ , \tag{3}$$

and write from (3) the over-determined system below:

$$\underbrace{\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix}}_{\boldsymbol{\omega}} = - \underbrace{\begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \\ I_{tn} \end{bmatrix}}_{\mathbf{b}} , \tag{4}$$

that can be easily solved using the least square method, such as:

$$\boldsymbol{\omega} = \left(A^T A\right)^{-1} A^T (-\mathbf{b}) \ .$$

The solution is optimal for all pixels in Ω_{ROI} and thus quite sensitive to any assumption violation:

- if Ω_{ROI} includes pixels which do not have the same motion than x,
- or if the considered image motion is too large regarding $|\Omega_{ROI}|$.

Such issues can be fixed, first by a prior investigation on the optimal size of $|\Omega_{ROI}|$, using synthetic data describing the same kind of motion than into the considered application (Fig. 1), and by applying a pyramidal coarse-to-fine scheme as described in [2].

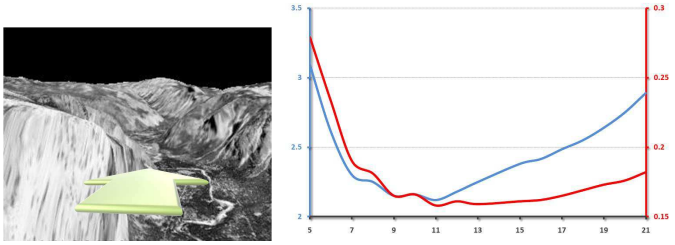

Figure 1. (Left) synthetic Yosemite video sequence where the green arrow shows the ego-motion of the virtual camera. (Right) the average norm and angular error of the Lucas Kanade optical flow, relatively to $|\Omega_{ROI}|$.

2) CUDA implementation: In case of vehicles operating in an open-environment, the perception process must be performed "on-the-fly", respecting as well a real-time constraint. However, current implementations of the pyramidal Lucas Kanade tracker do not perform in less than few seconds - depending of the CPU used - to provide a dense motion field on grayscale mid-resolution images (640×480 pixels). We proposed in [3] a real-time solution, using Nvidia CUDA technology, that executes the Lucas Kanade algorithm on a common GPU. This integration takes advantage of multi-processor architecture since optical flow can be computed independently for each pixel.

A comparison of the results obtained with our C++/CUDA library and the OpenCV version is presented Fig. 2, where the upper-left image corresponds to the first frame of the input video sequence. As showed in the upper-right image, the standard optical flow representation, that consists in a uniform distribution of 2-D vectors, does not allow visualizing efficiently the motion field smoothness. So we densly render the optical flow in the following illustrations with a color map coding the angle and the norm of each motion vector with respectively the hue and the brigthness pixel values.

By comparing with the OpenCV result (Fig. 2, second row), our implementation provides some interesting improvements, mainly due to the bilinear sampling of the motion field between two levels of the pyramid, instead of the nearest-neighbor interpolation used by OpenCV to reduce the computational time of its solution. Indeed, modern GPUs provide texture units able to perform this kind of bilinear interpolation without additional cost. The optical flow is computed 100 times faster with our method than on dual-core CPU, that is about 30 milliseconds.

B. Motion saliency and Tensor Voting

In some degenerated cases, such as textureless areas or occluded regions, the system (4) is not able to provide a good estimation of the image motion, and we need to compute a confident measure of the optical flow. However, since the least squares solution is optimal for each pixel into Ω_{ROI}, the optical flow constraint is supposed to be locally observed and thus, no photometric measurement can afford to evaluate the motion field previously computed. Nonetheless, by assuming

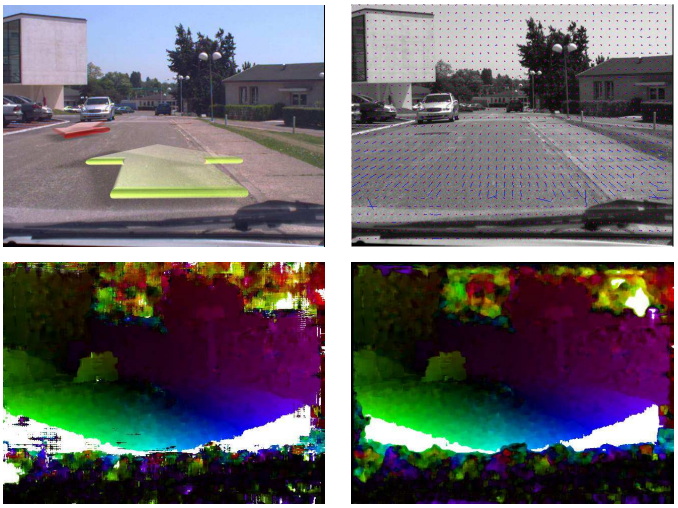

Figure 2. Results of the pyramidal approach of the Lucas and Kanade's optical flow algorithm, from the OpenCV implementation (left) and ours with CUDA (right). In addition of an execution 100 times faster, GPU allows to interpolate the motion field over-sampled at each level of the pyramid by hardware, avoiding the artifacts we can observe with the OpenCV version.

that disconuities only occur at 3-D object boundaries, we can assume the motion field globally smooth and so apply a spatial continuity constraint to check any wrong estimation. In this way, the following section describes use of 4-D Tensor Voting framework, initially presented by G. Medioni and M. Nicolescu [4], to quantitatively assess the optical flow.

1) 2-D Tensor Voting: Tensor Voting is a unified framework which has been developed by G. Medioni since the 90's. This formalism, based on tensor calculus for representing data and on a nonlinear voting process for their communication, identifies the geometrical structures described by the layout of a sparse and potentially noisy N-D dataset. It also provides for each point a saliency measure on its belonging to the structure inferred. As the N-D case is a generalization of the initial 2-D framework, we begin with this case before detailing the 4-D process.

In a 2-D space, Tensor Voting attempts to retrieve, from a set of sparse and isotropic data, geometrical structures such as curve elements and points. To do this, it uses the neighborhood layout of each considered input site to constrain their identification. Second order representation, in the form of a second order, symmetric, non-negative definite tensor, allows to encode the structural information of the input data, and to propagate this information to its neighborhood according to a simple tensor accumulation process.

Every 2-D tensor T can be represented using its quadratic form $\mathbf{T}$, that is a symmetric 2x2 matrix which also graphically defines an ellipse. According to the matrix algebra, $\mathbf{T}$ can be decomposed in the following way:

$$\mathbf{T} = (\lambda_1 - \lambda_2)\underbrace{\hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T}_{stick} + \lambda_2\underbrace{\left(\hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T + \hat{\mathbf{e}}_2\hat{\mathbf{e}}_2^T\right)}_{ball}. \quad (5)$$

where λ_i are the eigen values in decreasing order ($\lambda_i \leq \lambda_{i+1}$) and $\hat{\mathbf{e}}_i$ the corresponding unit eigen vectors. Each of the two terms corresponds to a specific elementary 2-D tensors: the first describes a degenerate elongated tensor and the second an isotropic one, named respectively stick and ball tensors. In 2-D Tensor Voting framework, data are encoded according to their isotropy with such unitary tensors. Thus, a curve element is coded by a stick tensor belonging to the normal of the considered curve while a point (without any structural information) is represented by a ball tensor.

The communication step is an accumulation process that involves a voter T_p and a votee (receiver) T_i. The voter infers on the votee according to the distance l between them. We voluntary limit the tensor voting scheme to its isotropic version, the ball vote, while the complete framework also takes into account the relative orientation between the voter and the votee. Each input site, originally coded by a unit ball tensor, communicates its relative position to its neighborhood using stick tensors that defines the normal direction to the straight line joining the votees. The stick is weighted with a Gaussian decay function that models its decreasing effect on its neighborhood with the distance l. Let $T_{i,t}$ be the tensor T_i at time t and $V_i\left(T_{j,t}\right)$ the vote it receives from a ball tensor $T_{j,t}$ of its neighborhood Ω. For each tensor T the voting process can be written as follows:

$$T_{p,2} = T_{p,1} + \sum_{i}^{\Omega} V_p\left(T_{i,1}\right) ,$$

where the quadratic form of V, that also corresponds to a tensor, is given by:

$$\mathbf{V} = e^{-\left(\frac{\|\mathbf{v}\|}{\sigma}\right)^2}\left(\mathbf{I} - \hat{\mathbf{v}}\hat{\mathbf{v}}^T\right) , \quad (6)$$

with $\mathbf{v}$ the vector joining the voter and the votee, and σ, a constant which defines the scope of the vote in the decay function. Extracting the structural information from the resulting tensors is done by decomposing them according to (5). The saliency measures $\lambda_1 - \lambda_2$ and λ_2 provide the preferred orientation of each tensor and so, the geometrical element it belongs to. As the method is non-iterative, it can be processed in a constant time complexity by parallel implementation.

2) Saliency measure in the joint-space (x, y, vx, vy): As introduced above, we suppose that the 3-D motion of any objects is continuous, as well as their motion in the focal plane, that just depends of the camera frame rate. Thus, the camera displacement provides a spatial collection of continuous motion areas in the focal plane that actually describe smooth surfaces in the 4-D joint-space (x, y, v_x, v_y) of image locations and motions. Fig. 3 illustrated such area in the 3-D sub-space (x, y, v_x). Using the Tensor-Voting-framework in 4-D, as described in [4], we propose to focus our attention on the discontinuities present in this space, that translate both area boundaries and wrong motion estimations.

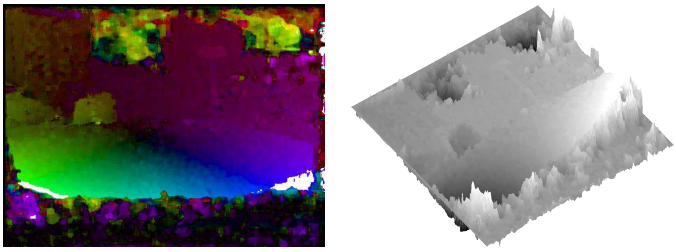

Figure 3. Representation of the sub-space (x, y, vx) (right) corresponding to the optical flow field illustrated by the left image. In such a space, road, as well as the car coming in our direction or the background, describe different smooth surfaces, in the opposite of the wrong motion estimation.

The N-dimensional Tensor Voting approach is an extension of the original framework, and the quadratic form of each 4-D tensor can also be decomposed in 4 elementary tensors as in (5):

$$\begin{aligned}\mathbf{T} &= (\lambda_1-\lambda_2)\underbrace{\hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T}_{stick} + (\lambda_2-\lambda_3)\underbrace{\left(\hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T+\hat{\mathbf{e}}_2\hat{\mathbf{e}}_2^T\right)}_{disk\,2D} \\ &+ (\lambda_3-\lambda_4)\underbrace{\left(\hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T+\hat{\mathbf{e}}_2\hat{\mathbf{e}}_2^T+\hat{\mathbf{e}}_3\hat{\mathbf{e}}_3^T\right)}_{disk\,3D} \\ &+ \lambda_4\underbrace{\left(\hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T+\hat{\mathbf{e}}_2\hat{\mathbf{e}}_2^T+\hat{\mathbf{e}}_3\hat{\mathbf{e}}_3^T+\hat{\mathbf{e}}_4\hat{\mathbf{e}}_4^T\right)}_{ball}\end{aligned} \quad (7)$$

where 2-D disk corresponds to the elementary tensor coding geometrical structure of type surface.

The first step consists in quantifying the relationship between the saliency measure obtained at the end of the Tensor Voting and the presence of noise into the input motion field previously computed. The graphic Fig. 4 displays the average norm and angular optical flow error relative to the normalized saliency value $\lambda_2 - \lambda_3$ computed on the synthetic video sequence Yosemite, with the optimal parameters found section II-A1. We can notice a threshold appears clearly around the value $\lambda_2 - \lambda_3 = 0.5$, after which it becomes impossible to distinguish motion discontinuities and noise. We use this value to threshold the motion field.

The bottom line Fig. 4 shows resulting motion field after applying this threshold on the surface saliency for our input video sequence. Smooth surfaces that correspond to continuous areas are preserved while almost all the noise is removed.

III. Depth map

We now assume that optical flow is estimated and refined as presented before, which includes computation of a reliability map corresponding to the saliency measure for each displacement vector. This section addresses the problem of extracting in real-time depth information from 2-D motion captured aboard a mobile architecture. We propose to robustly estimate the ground plane as a first step, before computing the parallax motion in an efficient planar parallax decomposition, and finally inferring the depth map. The ground is supposed locally planar.

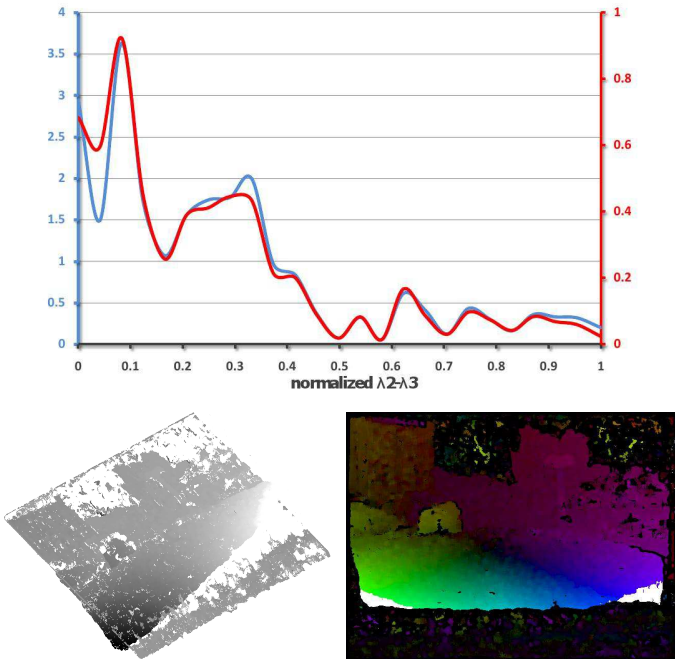

Figure 4. (Top) the average norm and angular motion field error relative to the surface saliency estimated by Tensor Voting, and the resulting optical flow after applying a threshold on the saliency value (bottom).

A. Robust and real-time ground estimation

To respect the real-time constraint, most of existing ground segmentation methods are feature-based. The general approach extracts road features and match them in two frames to compute the homography $\mathtt{H}$ induced by the road plane. Then, it identifies which image parts correspond to the ground, by computing a photometric error. This error is usually a function of the distance between the first image, warped according to the homographic transformation previously estimated, and the second one: $Err_{photo} = dist\left(I_2\left(\mathbf{x}\right), I_1\left(\mathtt{H}\mathbf{x}\right)\right)$. Such a distance is supposed to be null for all points lying on the ground and non determined otherwise. The result is actually only consistent at gradient discontinuities and very sensitive to homography estimation. The top-left image Fig. 5 illustrates such a photometric error using the l^2-norm on our video sequence: notice the noise on the road as well as poorly defined ground boundaries.

Since we can estimated the optical flow, we are also able to compute a more reliable error function, corresponding to the warping distance. It simply consists of the distance between two motion fields, the optical flow and the homographic transformation: $Err_{warp} = dist\left(LK\left(\mathbf{x}\right), \mathtt{H}\mathbf{x}\right)$. The top-right image in Fig. 5 shows the l^2-warping distance map obtained with the same homography matrix as used to compute photometric error (top-left image). The result can be interpreted and allows easily distinguishing the ground limits up to the horizon line. In addition, the measure is robust to thresholding, as illustrated on the graphic Fig. 5 where segmented road area appears stable despite a wide range of threshold values.

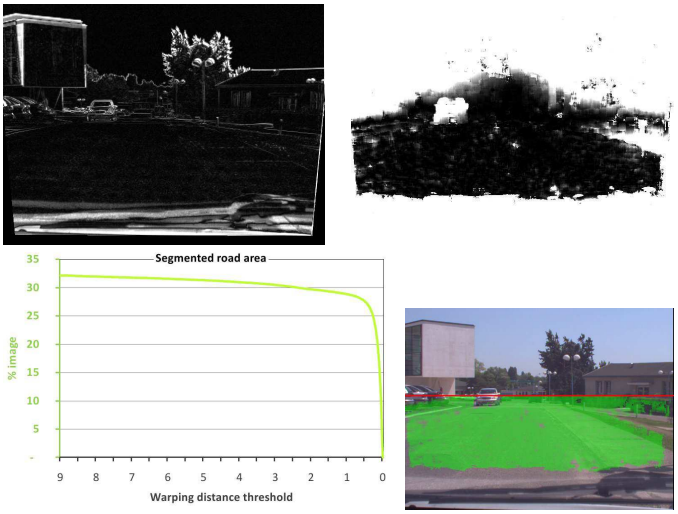

Figure 5. The warping distance (top-right) is more densly consistent than the photometric one (top-left). Also, it is robust to thresholding as illustrated by the area of the road which appears stable despite the wide range of threshold values (bottom line).

The homography H induced by the ground displacement between a couple of input frames is robustly computed with RANSAC, which performs a number of iterations that depends on the ratio of outliers in the image correspondences. We propose to use the Tensor Voting saliency map to select the best matches (regarding their motion continuity in the 4-D joint-space previously described) in each cell of a quantization grid, as RANSAC input. In this way, we can reduce the number of outliers corresponding to mismatched points and so, the number of iterations to estimate $\mathtt{H}$.

B. Absolute depth map

Following the general structure from motion scheme, we now infer depth map from the parallax motion. The parallax motion corresponds to the relative displacement of static objects in the focal plane, which is induced by the translational component of the ego-motion (a pure rotation does not provide any parallax). A way to estimate it consists in applying the "plane + parallax" motion decomposition that can be written as follows:

$$\boldsymbol{\omega} = \boldsymbol{\omega}_\pi + \boldsymbol{\mu}\,,$$

where $\boldsymbol{\omega}$, $\boldsymbol{\omega}_\pi$ and $\boldsymbol{\mu}$ denote respectively the image motion, the homographic transformation induced by an arbitrary plane of the scene (real or virtual) and the corresponding residual planar parallax motion. The rotational component of the camera displacement is enclosed in the homography and its subtraction from the optical flow provides a part of the translational image motion. In the literature, the homography of an arbitrary plane is used. That results in retrieving only a part of the parallel motion induced by the ego-motion, depending of the plane position. Instead, we compute the rotational homography that allows to estimate the full translational component nmerically more stable for the following steps.

Given the matrix $\mathtt{K}$ of camera intrinsic parameters, the homography $\mathtt{H}$ induced by a plane located at the distance d from the camera center and defined by its normal $\mathbf{n}$, is expressed by:

$$\mathtt{H} = \mathtt{K}\left(\mathtt{R} + \mathbf{t}\frac{1}{d}\mathbf{n}^T\right)\mathtt{K}^{-1} \tag{8}$$

where $\mathtt{R}$ and $\mathbf{t}$ are rotation and translation displacements of the camera between two views. Thus, the rotational homography corresponding to the image motion of point located at the infinite distance from the camera center, can be written as:

$$\mathtt{H}_{\mathtt{ROT}} = \mathtt{KRK}^{-1}\,. \tag{9}$$

The matrix $\mathtt{R}$ is here provided by decomposition of the road homography previously estimated (see [12] for more details).

Let $\mathbf{P}_i = (X, Y, Z)^T$ and $\mathbf{P}'_i = (X', Y', Z')^T$ denote the Cartesian coordinates of a scene point with respect to two different camera views, respectively. Let $\mathbf{p}_i = (x, y)^T$ and $\mathbf{p}'_i = (x', y')^T$ respectively denote the corresponding coordinates of the corresponding image points in the two image frames, and $\mathbf{t} = (t_X, t_Y, t_Z)^T$ the camera translation between the two views. If we note $\mathbf{p}_{w_i}$ the image point in the first frame which results from warping the corresponding point $\mathbf{p}'_i$ in the second image by the 2D transformation $\boldsymbol{\omega}_\pi$, when $t_z \neq 0$:

$$\boldsymbol{\omega}_\pi = (\mathbf{p}'_i - \mathbf{p}_{w_i})\,.$$

Also,

$$\boldsymbol{\mu} = \gamma\frac{t_Z}{d'_\pi}(\mathbf{e} - \mathbf{p}_{w_i})\,,$$

with $\mathbf{e}$ the epipole in the first frame, and d'_π the perpendicular distance from the second camera center to the reference plane. $\gamma = \dfrac{H}{Z}$ defines the projective structure where H is the perpendicular distance from $\mathbf{P}_i$ to the reference plane. For more details, see also [7]. The relative projective structure of two points P_1 and P_i is given by:

$$\frac{\gamma_i}{\gamma_1} = \frac{\boldsymbol{\mu}_i^T(\mathbf{p}_{w_i} - \mathbf{p}_{w_1})_\perp}{\boldsymbol{\mu}_1^T(\mathbf{p}_{w_i} - \mathbf{p}_{w_1})_\perp}\,, \tag{10}$$

that corresponds to the relative depth of $\mathbf{P}_i$ regarding $\mathbf{P}_1$. It allows computing a relative depth map of the scene. However, since the the distance of camera to the ground is supposed to be constant and measured off-line, the absolute depth can be computed for any image point p_i by choosing P_1 lying on the road (but not necessary in the camera field of view). The left image Fig. 6 shows the depth map computed with P_1 chosen such as its projection p_1 would be located at the bottom-right corner of the image. Note the singular line in the direction of the parallax displacement of the reference point, when $\boldsymbol{\mu}_1^T(\mathbf{p}_{w_i} - \mathbf{p}_{w_1})_\perp$ tends to 0. It can be fixed by adding a second reference point P_2 if its corresponding image point p_2 is far enough from the singular line. Also, the relationship of the relative projective structure regarding P_1 and P_2 is given by: $\gamma_i/\gamma_1 = \gamma_2/\gamma_1 \cdot \gamma_i/\gamma_2$. In the right image Fig. 6, p_1 and p_2 correspond to the right and left bottom corner, respectively.

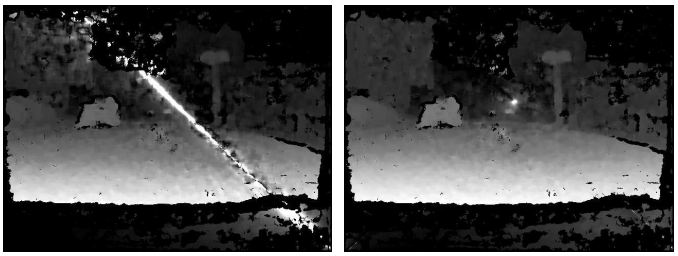

Figure 6. Relative projective structure of every image point, computed regarding only one reference point (left), and two reference points (right).

IV. Optimizations

In case of large motions or in presence of severe occlusions, the optical flow can be mistaken. The current section introduces some improvement strategies to overcome this kind of issues, by taking advantage of the approach presented above. The first optimization consists in combining several complementary motion fields, by using Tensor Voting framework to choose at each pixel, which one provides the most relevant displacement. The second one addresses the problem of initializing the optical flow.

A. Multi-flow Tensor Voting

Even if it provides a reliable solution in almost cases, the pyramidal approach of the Lucas and Kanade algorithm actually induced some issues. Due to the relative aperture size in highest levels of the pyramid, the original assumption consisting in a uniform velocity into the neighborhood Ω, is often violated close to the object boundaries. It implies a wrong initialization of lower levels, especially in presence of occlusions. Also, the pyramidal approach forbids to initialize the optical flow algorithm with a prior motion field. Again, the relative size of the aperture is too large in the highest level regarding to the norm of an initial down-sampled motion field. On an other hand, the original algorithm performed on only one level is well-adapted to be initialized by a prior motion field when the image motion is continuous, since the aperture size would be consistent with the norm of the initial motion at each pixel. So we propose to combine both the advantage of the pyramidal and non-pyramidal approaches.

Let consider the 4-D input space detailed section II-B2 containing n different motion fields. At each pixel location corresponds n velocity vectors. By performing the voting stage as described in II-B1, each tensor provides its relative position to all its neighbors that do not belong to the same image location. The final motion field is simply built by selecting, at each pixel, the velocity vector corresponding to the tensor with the higher surface saliency value.

B. Motion field filtering

The second optimization concerns initialization of optical flow as mentioned above. Assuming that camera displacement is continuous, we can simply use the result of our previous iteration as initialization for the current estimation of optical flow on one level, and combine it with pyramidal optical flow. The result is illustrated Fig.8 with the bottom-left image, while the first input frame and the depth map computed without any optimization are given in the first row. Even if we can observed some improvements, especially in the bottom of the map, the depth is still not correctly evaluated in some large ground areas that are too close to image parts occluded due to the ego-motion, or not evaluated when motion field has been previously filtered due to discontinuities.

Figure 7.

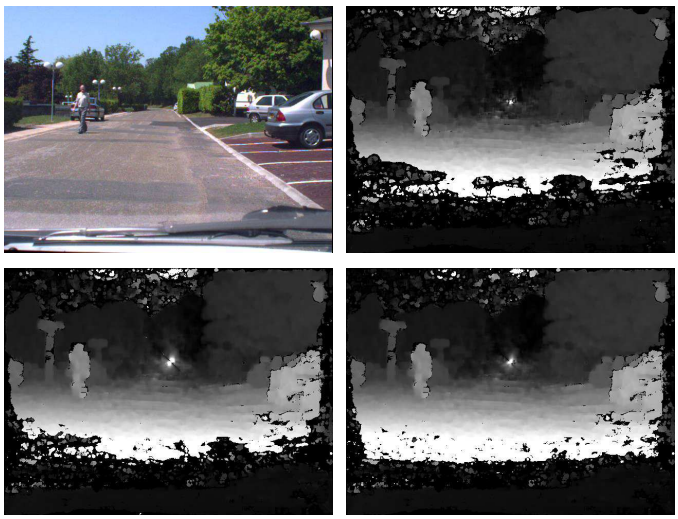

Figure 8. Depth map computed with different level of optimization (see text for details).

To initialize optical flow algorithm with a dense motion field, despite the threshold applied at the end of the Tensor Voting process, we propose to replace every missing velocity vectors with the corresponding homographic transformation induced by the ground plane under the horizon. We also proceed in the same manner above the horizon, by using the rotational homographic transformation. The result is shown in the bottom-right image Fig.8: the road depth appears clearly better estimated up to the occluded area boundaries. Note that the depth map Fig.6 includes these improvements and also presents a clear demarcation between the road and the occluded regions.

V. Conclusion

We presented a two-frames depth map estimation process, as well as its partial GPU integration that allows real-time constraints. The overall approach, with the optimization steps, is performed at about 12 frames per second for on mid-resolution images (640×480 pixels). The hardware configuration includes a 448 Cuda core architecture (GPU) combined with a dual core processor (CPU). It is easily scalable to the

next generation of architecture to increase both speed and accuracy.

Because the planar parallax constraints are directly inferred by the epipolar ones, the depth maps we proposed are only relevant for static objects, and we plan to investigate the way to extend the 3-D recovering to mobile objects too. Also, we will focus on the integration on these results over more than two frames, and improve the optical flow filtering.

REFERENCES

[1] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", in Proceeding on Image Understanding workshop, pp. 121-130, 1981.

[2] J. Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm", in Technical Report, Intel Corporation, Microsoft Research Labs, 2000.

[3] J. Marzat, Y. Dumortier, and A. Ducrot, "Real-Time Dense and Accurate Parallel Optical Flow using CUDA", in 17th WSCG International Conference on Computer Graphics, Visualization (WSCG'09), 2009.

[4] M. Nicolescu and G. Medioni, "Motion Segmentation with Accurate Boundaries - A Tensor Voting Approach", in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1(1):382, 2003.

[5] A. Talukder, S. Goldberg, L. Matthies and A. Ansar, "Real-time detection of moving objects in a dynamic scene from moving robots vehicles", in IEEE International Conference on Intelligent Robots and Systems (IROS'03), 2003.

[6] A. Ess, B. Leibe, K. Schindler and L. Van Gool, "Moving Obstacle Detection in Highly Dynamic Scenes", in International Conference on Robotics and Automation (ICRA'09), Kobe, Japan, May 2009.

[7] M. Irani and P. Anandan, "A unified approach to moving object detection in 2d and 3d scenes", in IEEE Trans. Pattern Analysis & Machine Intelligence (TPAMI'98), 20(6):577-589, June 1998.

[8] Guofeng Zhang, Jiaya Jia, Tien-Tsin Wong and Hujun Bao, "Consistent Depth Maps Recovery from a Video Sequence", in IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI'09), 31(6):974-988, 2009.

[9] E. Eade and T. Drummond, "Edge landmarks in monocular SLAM", in Image Vision Computing (IVC'09), 27(5):588-596, 2009.

[10] Y. Dumortier, I. Herlin and A. Ducrot, "4-D Tensor Voting Motion Segmentation for Obstacle Detection in Autonomous Guided Vehicle", in Intelligent Vehicles Symposium (IV'08), pp. 379-384, 2008.

[11] A. Wedel, D. Cremers, T. Pock and H. Bischof, "Structure and Motion-adaptive Regularization for High Accuracy Optic Flow", in IEEE International Conference on Computer Vision (ICCV'09)}, Kyoto, Japan, 2009.

[12] R. J. Hartley. and A. Zisserman, "Multiple View Geometry in Computer Vision", Cambridge University Press, 2000.